



Acquisition Reform: Leadership in Balancing Cost, Schedule and Performance

The Fall 2011 Software Assurance Forum
Software Engineering Institute
Arlington, VA 22203
12-16 September 2011

Dr. Kenneth E. Nidiffer
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
703-908-1117



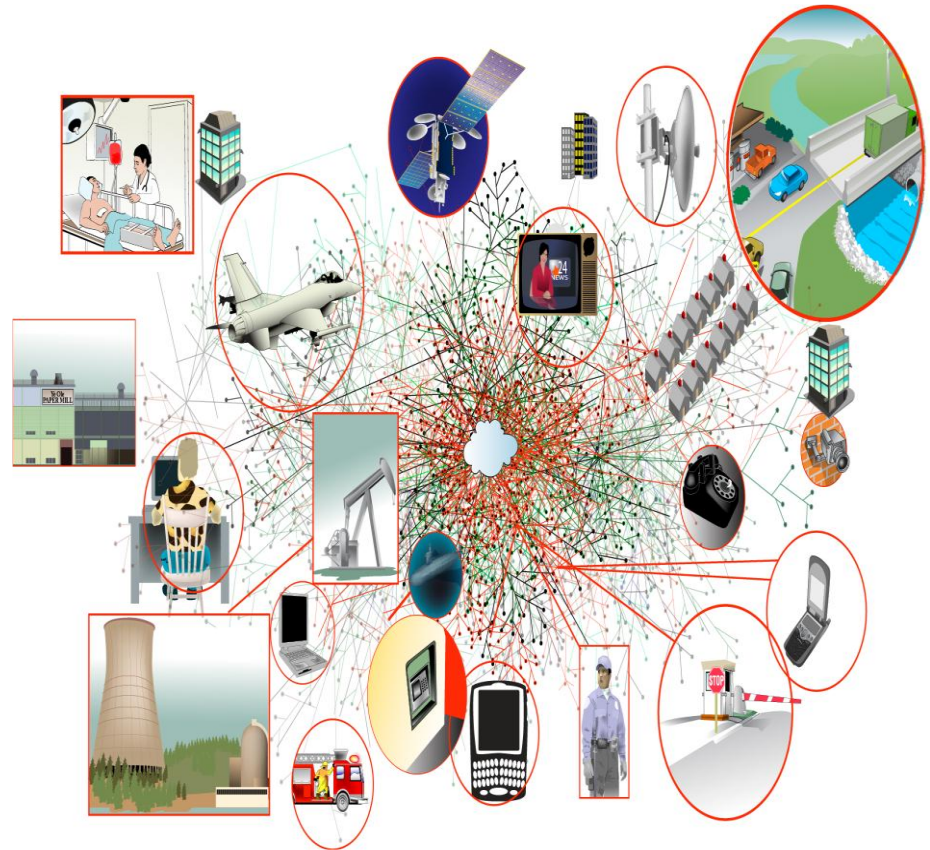
Software Engineering Institute

Carnegie Mellon

© 2011 Carnegie Mellon University

Overview

- Introductions
- Panel Discussions
- Q&A Session



Our civilization runs on software

*Bjarne Stroustrup**

* Danish computer scientist, most notable for the creation and the development of the widely used C++



Software Assurance

Software assurance (SwA: “the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner”*)

Software assurance addresses:

- Trustworthiness - No exploitable vulnerabilities exist, either maliciously or unintentionally inserted;
- Predictable Execution - Justifiable confidence that software, when executed, functions as intended;
- Conformance - Planned and systematic set of multi-disciplinary activities that ensure software processes and products conform to requirements, standards/procedures.

*Source: National Information Assurance Glossary"; CNSS Instruction No. 4009 [National Information Assurance Glossary](#)

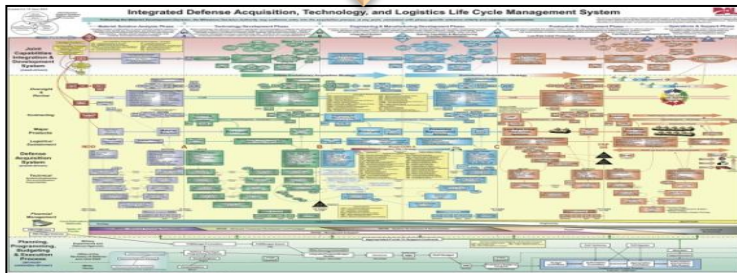


Acquisition: IT is different from a Weapon System --- and Critical to Enable a More Resilient Cyber Environment

Weapon Systems



- Weapon platform centric
- Military unique requirements
- Development of military-unique, breakthrough technologies
- Development cycle of decade or more
- Production decisions for unique HW
- Service lives extending into decades



IT Systems



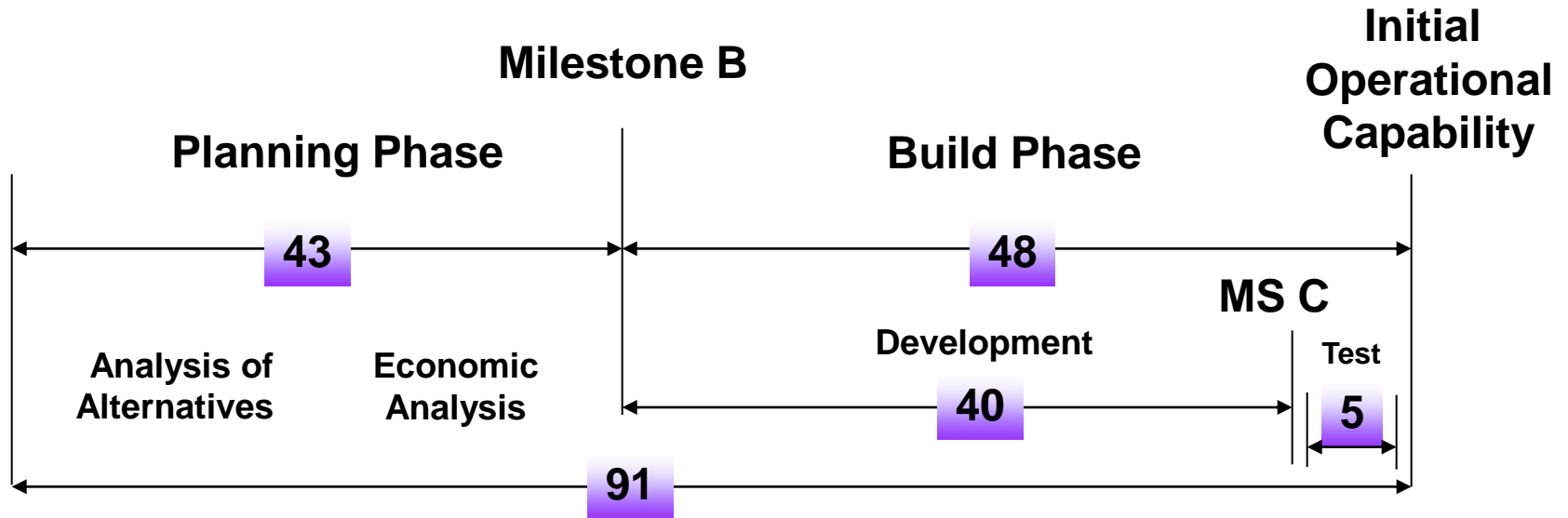
- Enterprise network centric
- Adapt commercial capabilities for military needs
- Leverage commercial technologies
- Technology cycle 12-18 months
- Procure commodity HW
- Periodic technology refresh to avoid obsolescence



Demands a Different Acquisition Process



DoD IT Acquisition Cycle-Time - 32 MAIS



*Cycle-Time Driven by Processes Developed to Counter
a Cold War Adversary In Industrial Age Society*

Source: Defense Science Board Report, March 2009



IT Acquisition Reform Imperative

Congress

- Develop and Implement a new process for Acquiring IT (FY10 NDAA* Section 804)
- HASC** Panel on Defense Acquisition Reform Finding and Recommendations (23 March 2010)

Widely documented Problems with DoD IT Acquisitions

- Defense Science Board
 - Jan '09 – Integrating COTS
 - Mar '09 – IT Acquisition
 - Apr '09 – Fix the Acq process
 - Jul '09 – Rapid Acquisition
- Industry Associations
 - AFEI, TechAmerica,
- National Academies - Achieving Effective Acq of IT in DoD 2010
- Business Leads – Aug '08 Joint DISA IT Review



Federal CIO

25-Pt Implementation Plan to Reform Federal IT Management
Vivek Kundra, U.S. CIO, December 9, 2010

DoD Senior Leadership Vision

“First step [for DoD to succeed in delivery of IT] is to acknowledge that simply tailoring the existing processes in not sufficient” (National Research Council, DEC 2009)

NDAA: National Defense Authorization Act ; HASC: House Armed Services Committee;
AFEI: Association for Enterprise Information; DISA: Defense Information Systems Agency

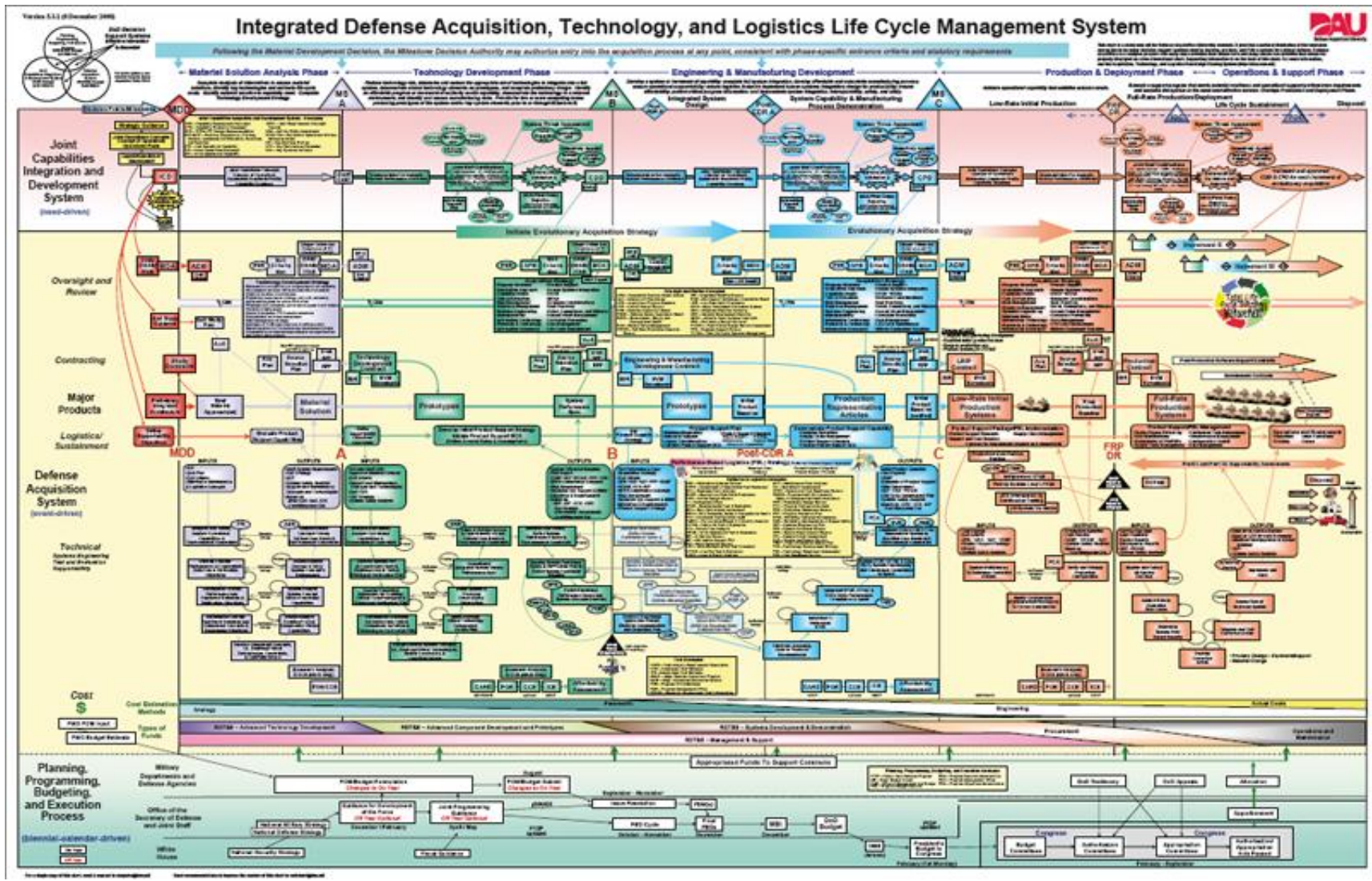


DoD IT Acquisition Challenges

	House Armed Services Committee	National Research Council	Defense Science Board	Business Executives for National Security
Defense acquisition process structured for weapon systems; ill-suited for information technology	✓	✓	✓	✓
Systems take too long to deliver; inconsistent with technology cycle	✓	✓	✓	
Too document intensive, time consuming, and process bound to respond effectively to end-user needs	✓	✓	✓	✓
Oversight process not aligned with rapid acquisitions (favors large programs, high-level oversight)		✓		✓
Lack of accountability by personnel in the oversight process		✓		✓
Complexity inherent in aligning three major Departmental processes - Requirements, Resourcing and Acquisition	✓			✓
Funding process inconsistent with pace of evolving mission requirements	✓	✓		
Current metrics (financial, acquisition process) don't work well in measuring IT success	✓	✓		
Lack of meaningful trades between performance, cost, and date-to-field	✓	✓	✓	✓
Overly detailed requirements that are inconsistent with pace of technology change and need for rapid delivery	✓	✓		✓
Inability to prioritize requirements effectively	✓	✓		✓
Testing is integrated too late and serially	✓	✓		
Cyber-security is inadequately managed during the acquisition process			✓	
Lack sufficient numbers of individuals with proven records of acquisition success	✓	✓	✓	✓
Significant cultural impediments to change	✓			✓



An Effective Process for Major Defense Systems – Considers Complete Life Cycle – Not Very Agile



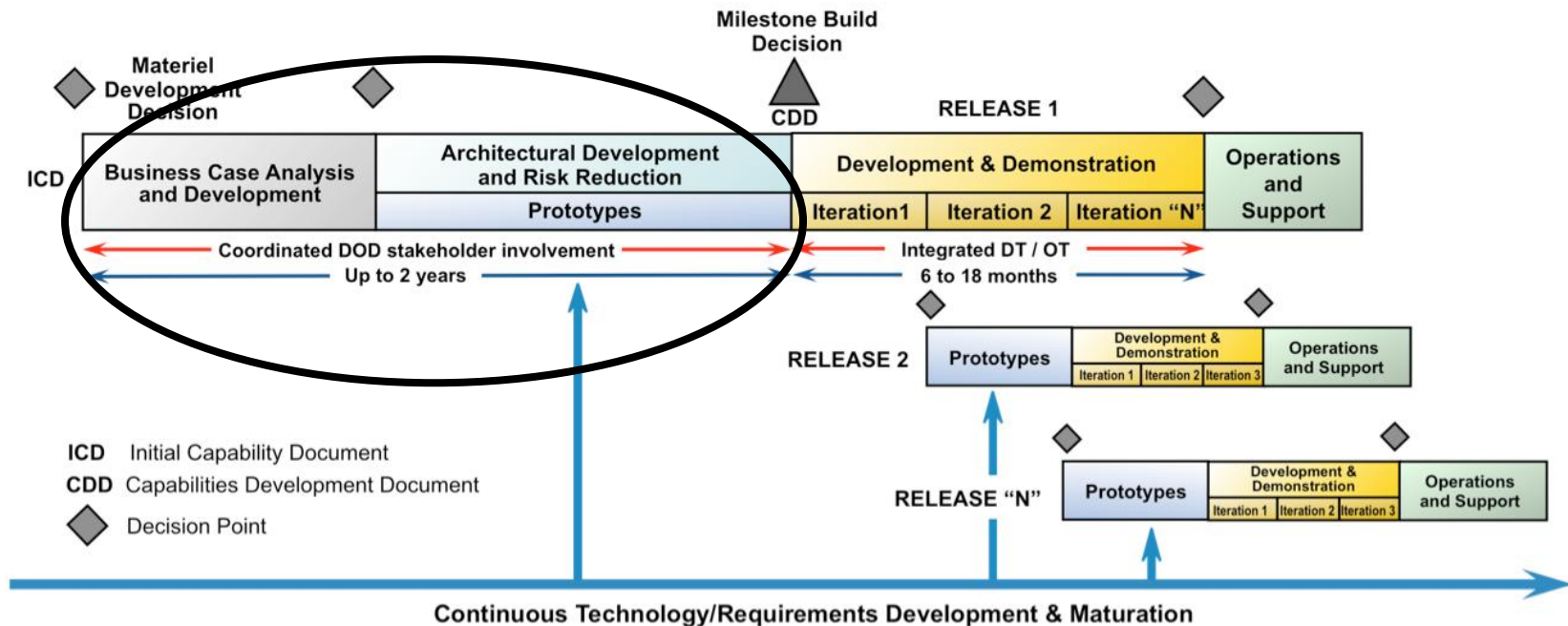
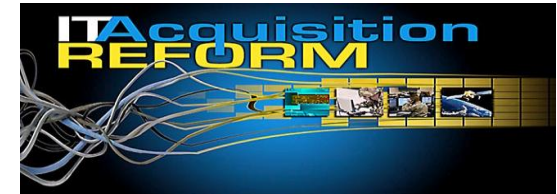
Software Evolution and Maintenance Cost Is Increasing

Year	Proportion of software maintenance costs	Definition	Reference
2000	>90%	Software cost devoted to system maintenance & evolution / total software costs	Erlikh (2000)
1993	75%	Software maintenance / information system budget (in Fortune 1000 companies)	Eastwood (1993)
1990	>90%	Software cost devoted to system maintenance & evolution / total software costs	Moad (1990)
1990	60-70%	Software maintenance / total management information systems (MIS) operating budgets	Huff (1990)
1988	60-70%	Software maintenance / total management information systems (MIS) operating budgets	Port (1988)
1984	65-75%	Effort spent on software maintenance / total available software engineering effort.	McKee (1984)
1981	>50%	Staff time spent on maintenance / total time (in 487 organizations)	Lientz & Swanson (1981)
1979	67%	Maintenance costs / total software costs	Zelkowitz <i>et al.</i> (1979)

Source: [Jussi Koskinen](#), Department of Computer Science and Information Systems, University of Jyväskylä
P.O. Box 35, 40014 Jyväskylä, Finland



Accelerating Assured Software Delivery for the Mission



Front-end systems and software engineering are critical disciplines of delivering enhance incremental software capabilities



Why Are Software Intensive IT Projects Difficult?

According to Fred Brooks* software projects are difficult because of accidental and essential difficulties

- Accidental difficulties are caused by the current state of our understanding
 - of methods, tools, and techniques
 - of the underlying technology base
- Essential difficulties are caused by the inherent nature of software
 - invisibility - lack of physical properties
 - conformity
 - changeability
 - complexity

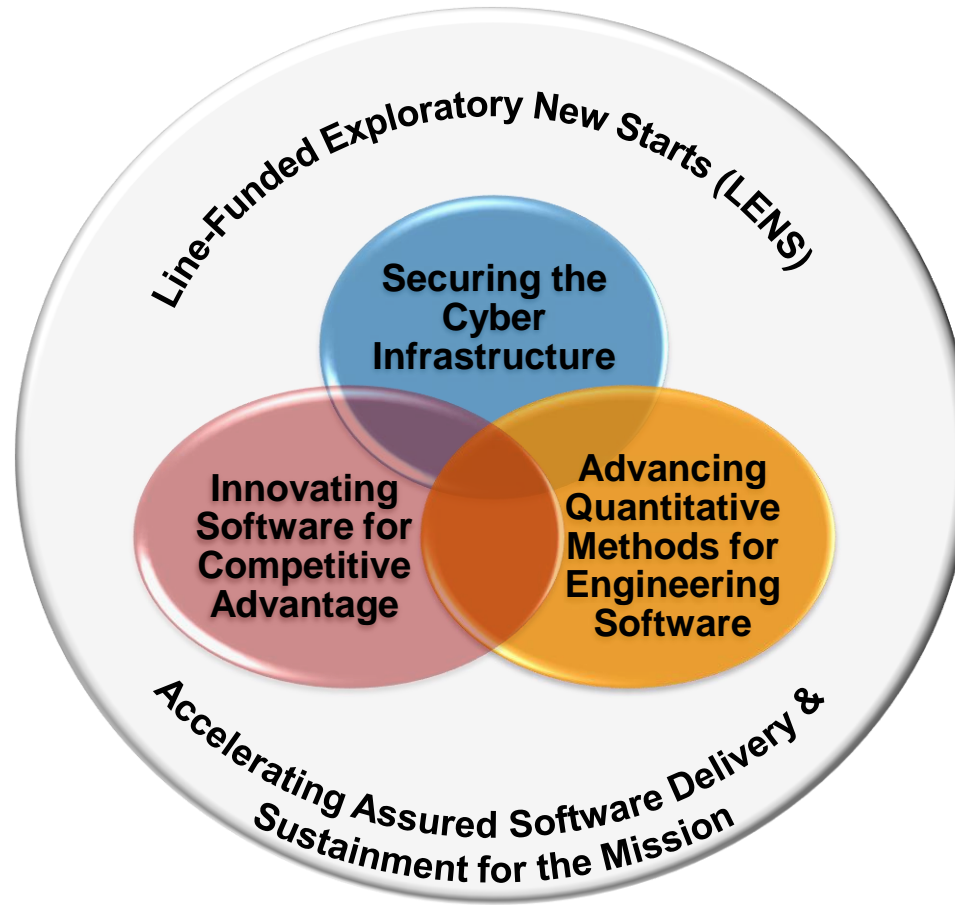


Dr. Fred Brooks, Jr.

* Source: *The Mythical Man-Month* by Fred Brooks, Addison Wesley, 1995



Accelerating Assured Software Delivery for the Mission



The Master of Software Assurance (MSwA) Reference Curriculum

<http://www.cert.org/mswa/>

EXPLORE

CREATE

APPLY

AMPLIFY

SUSTAIN



Software Engineering Institute

Carnegie Mellon

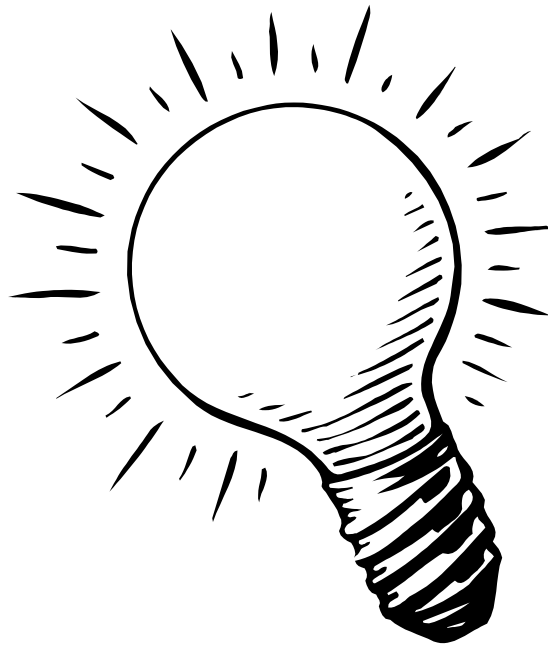
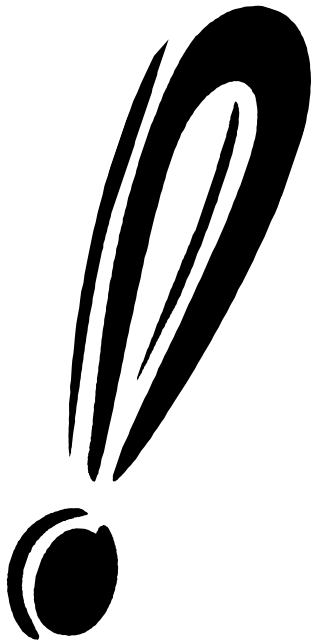
The Fall 2011 Software Assurance Forum
Dr. Kenneth E. Nidiffer, September 2011
© 2011 Carnegie Mellon University



**Homeland
Security**



Will Be Addressed In Q&A Session



Software Engineering Institute

Carnegie Mellon

The Fall 2011 Software Assurance Forum
Dr. Kenneth E. Nidiffer, September 2011
© 2011 Carnegie Mellon University

Contact Information

Dr. Kenneth E. Nidiffer, Director of Strategic Plans for Government Programs

Software Engineering Institute, Carnegie Mellon University

Office: + 1 703-908-1117

Fax: + 1 703-908-9317

Email: nidiffer@sei.cmu.edu



References

1. Eastwood, A. (1993). "Firm fires shots at legacy systems". *Computing Canada* **19** (2), p. 17.
2. Erlikh, L. (2000). "Leveraging legacy system dollars for E-business". *(IEEE) IT Pro*, May/June 2000, 17-23.
3. Huff, S. (1990). "Information systems maintenance". *The Business Quarterly* **55**, 30-32.
4. Lientz, B.P. & Swanson, E. (1980). "*Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations*". Addison-Wesley: Reading, MA, 214 p.
5. Lientz, B.P. & Swanson, E. (1981). "Problems in application software maintenance". *Communications of the ACM* **24** (11), 763-769.
6. Martin, J. (1983). "*Software Maintenance: The Problem and Its Solution*". Prentice Hall, 472 p.
7. McKee, J. (1984). "Maintenance as a function of design". *Proceedings of the AFIPS National Computer Conference*, 187-193.
8. Moad, J. (1990). "Maintaining the competitive edge". *Datamation* 61-62, 64, 66.
9. Nosek, J. & Palvia, P. (1990). "Software maintenance management: changes in the last decade". *Journal of Software Maintenance: Research and Practice* **2** (3), 157-174.
10. Port, O. (1988). "The software trap – automate or else". *Business Week* **3051** (9), 142-154.
11. Sommerville, I. (2000). "*Software Engineering (6th Edition)*". Addison-Wesley.
12. Standish, T. (1984). "An essay on software reuse". *IEEE Transactions on Software Engineering* **SE-10** (5), 494-497.
13. Ulrich, W. (1990). "The evolutionary growth of software engineering and the decade ahead". *American Programmer* **3** (10), 12-20.
14. Zelkowitz, M., Shaw, A. & Gannon, J. (1979). "*Principles of Software Engineering and Design*". Prentice-Hall.



NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

